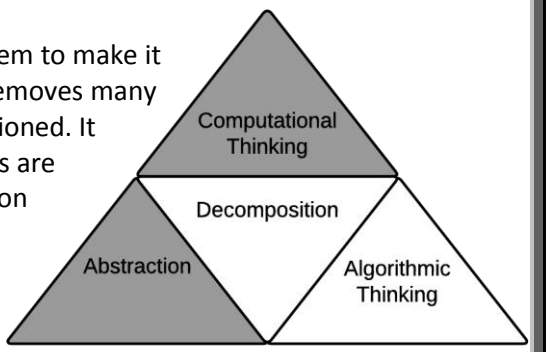
Abstraction - Reading



Abstraction is all about removing unnecessary detail from a problem to make it easier to solve. For instance, a map of the London Underground removes many details about the stations, tracks and exactly where they are positioned. It contains the most important information about where the stations are and the different lines have connection points. This is an abstraction which removes all the unnecessary details that are needed for planning a journey.

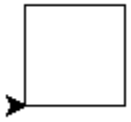
When we develop solutions to problems, often we are trying to find patterns, similarities and connections between different problems. This way we can solve new problems using existing solutions. The process of trying to create one solution to a problem which can then solve many different problems is called **generalisation**.

For instance, we could teach a child how to solve the problem of making a cake by teaching general skills like reading a recipe and mixing. This would be teaching the abstract concepts and ignoring the details of any specific cake. If the child understands how to bake the cake in an abstract way then they will be able to bake all kinds of cakes, such as a sponge cake, fruit cake or carrot cake. By using an abstraction we do not need to worry about the details of the problem such as the exact weight of the flour or the exact time it needs to bake in the oven.

This is very important for **computational thinking** as we want to try to create general solutions to problems which can be reused in many different ways. Now let's consider a real program. The following program is written in Python in two different ways. Both ways produce a square.

```
import turtle

wn = turtle.Screen()
sam = turtle.Turtle()

sam.forward(50)
sam.left(90)
sam.forward(50)
sam.left(90)
sam.forward(50)
sam.left(90)
sam.forward(50)
sam.left(90)

wn.mainloop()
```

```
import turtle

def drawPolygon(turtle, sides, length):
    for i in range(sides):
        turtle.forward(length)
        turtle.left(360/sides)

wn = turtle.Screen()
sam = turtle.Turtle()
drawPolygon(sam, 4, 50)

wn.mainloop()
```



The code on the left is not abstract. It creates a turtle called *sam* and then tells it exactly how far to move forward and exactly how many degrees to turn left. The program on the right creates an abstract **procedure**. This can be **reused** for any turtle, with any size of polygon. This time we tell the computer general rules such as *the amount to turn left is 360 degrees divided by the number of sides*. This is an example of generalisation. Whenever you feel like copying and pasting code it should be a warning to you that there is probably a better abstract solution. Drawing a seven-sided polygon is now very easy and needs just one line of code:

```
drawPolygon(sam, 7, 20)
```

Creating abstract **subroutines** is called **control abstraction** as we abstract the **control structures** used. **Programming languages** use **data types**, which is a **data abstraction**. Rather than dealing with **binary** (ones and zeros) for numbers we deal with **base 10 numbers**.